|  |  |
| --- | --- |
| **Ex.No:** 8 | **Implementation of File Transfer Protocol** |
| **Date:** |

**Aim:** To write a program in java to implement file transfer between client and file server.

**Algorithm**:

1. Create a socket connection from client to the server with corresponding IP address and port number

2. Create necessary file streams

3. Get the option from the user to SEND/ DISCONNECT and the file name to be Transferred

4. If SEND, read each byte from the file in the client side and write it into the output stream of the socket

5. In the server side, read the bytes from the input stream and write it into the file

**Server**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class FTPServer

{

public static void main(String args[]) throws Exception

{

ServerSocket soc=new ServerSocket(5217);

System.out.println("FTP Server Started on Port Number 5217");

while(true)

{

System.out.println("Waiting for Connection ...");

transferfile t=new transferfile(soc.accept());

}

}

}

class transferfile extends Thread

{

Socket ClientSoc;

DataInputStream din;

DataOutputStream dout;

transferfile(Socket soc)

{

try

{

ClientSoc=soc;

din=new DataInputStream(ClientSoc.getInputStream());

dout=new DataOutputStream(ClientSoc.getOutputStream());

System.out.println("FTP Client Connected ...");

start();

}

catch(Exception ex)

{

}

}

void SendFile() throws Exception

{

String filename=din.readUTF();

File f=new File(filename);

if(!f.exists())

{

dout.writeUTF("File Not Found");

return;

}

else

{

dout.writeUTF("READY");

FileInputStream fin=new FileInputStream(f);

int ch;

do

{

ch=fin.read();

dout.writeUTF(String.valueOf(ch));

}

while(ch!=-1);

fin.close();

dout.writeUTF("File Receive Successfully");

}

}

void ReceiveFile() throws Exception

{

String filename=din.readUTF();

if(filename.compareTo("File not found")==0)

{

return;

}

File f=new File(filename);

String option;

if(f.exists())

{

dout.writeUTF("File Already Exists");

option=din.readUTF();

}

else

{

dout.writeUTF("SendFile");

option="Y";

}

if(option.compareTo("Y")==0)

{

FileOutputStream fout=new FileOutputStream(f);

int ch;

String temp;

do

{

temp=din.readUTF();

ch=Integer.parseInt(temp);

if(ch!=-1)

{

fout.write(ch);

}

}while(ch!=-1);

fout.close();

dout.writeUTF("File Send Successfully");

}

else

{

return;

}

}

public void run()

{

while(true)

{

try

{

System.out.println("Waiting for Command ...");

String Command=din.readUTF();

if(Command.compareTo("GET")==0)

{

System.out.println("\tGET Command Received ...");

SendFile();

continue;

}

else if(Command.compareTo("SEND")==0)

{

System.out.println("\tSEND Command Received ...");

ReceiveFile();

continue;

}

else if(Command.compareTo("DISCONNECT")==0)

{

System.out.println("\tDisconnect Command Received ...");

System.exit(1);

}

}

catch(Exception ex)

{

}

}

}

}

import java.net.\*;

import java.io.\*;

import java.util.\*;

class FTPClient

{

public static void main(String args[]) throws Exception

{

Socket soc=new Socket("127.0.0.1",5217);

transferfileClient t=new transferfileClient(soc);

t.displayMenu();

}

}

class transferfileClient

{

Socket ClientSoc;

DataInputStream din;

DataOutputStream dout;

BufferedReader br;

transferfileClient(Socket soc)

{

try

{

ClientSoc=soc;

din=new DataInputStream(ClientSoc.getInputStream());

dout=new DataOutputStream(ClientSoc.getOutputStream());

br=new BufferedReader(new InputStreamReader(System.in));

}

catch(Exception ex)

{

}

}

void SendFile() throws Exception

{

String filename;

System.out.print("Enter File Name :");

filename=br.readLine();

File f=new File(filename);

if(!f.exists())

{

System.out.println("File not Exists...");

dout.writeUTF("File not found");

return;

}

dout.writeUTF(filename);

String msgFromServer=din.readUTF();

if(msgFromServer.compareTo("File Already Exists")==0)

{

String Option;

System.out.println("File Already Exists. Want to OverWrite (Y/N) ?");

Option=br.readLine();

if(Option=="Y")

{

dout.writeUTF("Y");

}

else

{

dout.writeUTF("N");

return;

}

}

System.out.println("Sending File ...");

FileInputStream fin=new FileInputStream(f);

int ch;

do

{

ch=fin.read();

dout.writeUTF(String.valueOf(ch));

}

while(ch!=-1);

fin.close();

System.out.println(din.readUTF());

}

void ReceiveFile() throws Exception

{

String fileName;

System.out.print("Enter File Name :");

fileName=br.readLine();

dout.writeUTF(fileName);

String msgFromServer=din.readUTF();

if(msgFromServer.compareTo("File Not Found")==0)

{

System.out.println("File not found on Server ...");

return;

}

else if(msgFromServer.compareTo("READY")==0)

{

System.out.println("Receiving File ...");

File f=new File(fileName);

if(f.exists())

{

String Option;

System.out.println("File Already Exists. Want to OverWrite (Y/N) ?");

Option=br.readLine();

if(Option=="N")

{

dout.flush();

return;

}

}

FileOutputStream fout=new FileOutputStream(f);

int ch;

String temp;

do

{

temp=din.readUTF();

ch=Integer.parseInt(temp);

if(ch!=-1)

{

fout.write(ch);

}

}while(ch!=-1);

fout.close();

System.out.println(din.readUTF());

}

}

public void displayMenu() throws Exception

{

while(true)

{

System.out.println("[ MENU ]");

System.out.println("1. Send File");

System.out.println("2. Receive File");

System.out.println("3. Exit");

System.out.print("\nEnter Choice :");

int choice;

choice=Integer.parseInt(br.readLine());

if(choice==1)

{

dout.writeUTF("SEND");

SendFile();

}

else if(choice==2)

{

dout.writeUTF("GET");

ReceiveFile();

}

else

{

dout.writeUTF("DISCONNECT");

System.exit(1);

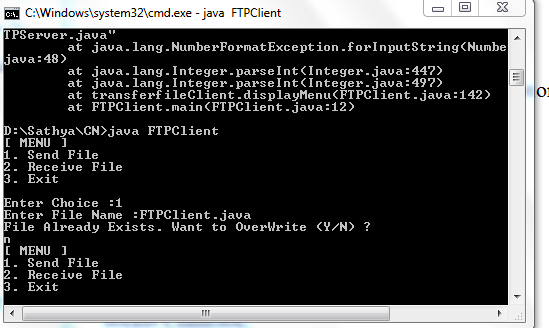
}

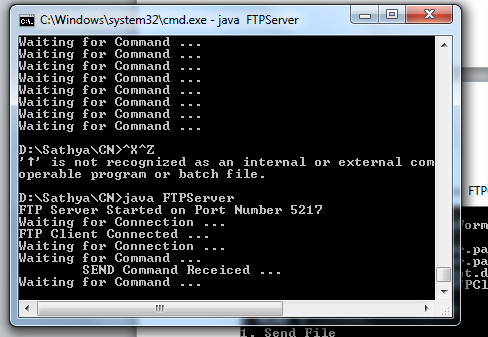
}

}

}

**OUTPUT**





**RESULT:**

Thus FTP using TCP has been implemented using JAVA Socket Programming

|  |  |
| --- | --- |
| **Ex.No:** 9 | **Remote Command Execution Using UDP** |
| **Date:** |

**Aim:**  To execute commands remotely using UDP

**Algorithm**

**Client-Side Programming**

Open the socket connection

Communication: In the communication part, there is a slight change. The difference with the previous article lies in the usage of both the input and output streams to send commands and receive the results to and from the server respectively. DataInputStream and DataOutputStream are used instead of basic InputStream and OutputStream to make it machine independent.

**Server-Side Programming**

Steps involved on the server side are as follows-

Establish a socket connection.

Process the equations coming from client: In server side also we open both the inputStream and outputStream. After receiving the equation, we process it and return the result back to the client by writing on the outputStream of the socket.

Close the connection.

**Program**

**Client**

import java.io.\*;

import java.net.\*;

class RemoteClient

{

public static void main(String args[])

{

try

{

int Port;

BufferedReader Buf =new BufferedReader(new

InputStreamReader(System.in));

System.out.print(" Enter the Port Address : " );

Port=Integer.parseInt(Buf.readLine());

Socket s=new Socket("localhost",Port);

if(s.isConnected()==true)

System.out.println(" Server Socket is Connected Successfully. ");

InputStream in=s.getInputStream();

OutputStream ou=s.getOutputStream();

BufferedReader buf=new BufferedReader(new

InputStreamReader(System.in));

BufferedReader buf1=new BufferedReader(new

InputStreamReader(in));

PrintWriter pr=new PrintWriter(ou);

System.out.print(" Enter the Command to be Executed : " );

pr.println(buf.readLine());

pr.flush();

String str=buf1.readLine();

System.out.println(" " + str + " Opened Successfully. ");

System.out.println(" The " + str + " Command is Executed Successfully. ");

pr.close();

ou.close();

in.close();

}

catch(Exception e)

{

System.out.println(" Error : " + e.getMessage());

}

}

}

**Server**

import java.io.\*;

import java.net.\*;

class RemoteServer

{

public static void main(String args[])

{

try

{

int Port;

BufferedReader Buf =new BufferedReader(new

InputStreamReader(System.in));

System.out.print(" Enter the Port Address : " );

Port=Integer.parseInt(Buf.readLine());

ServerSocket ss=new ServerSocket(Port);

System.out.println(" Server is Ready To Receive a Command. ");

System.out.println(" Waiting ..... ");

Socket s=ss.accept();

if(s.isConnected()==true)

System.out.println('' Client Socket is Connected Successfully. ");

InputStream in=s.getInputStream();

OutputStream ou=s.getOutputStream();

BufferedReader buf=new BufferedReader(new

InputStreamReader(in));

String command=buf.readLine();

PrintWriter pr=new PrintWriter(ou);

pr.println(command);

Runtime H=Runtime.getRuntime();

Process P=H.exec(command);

System.out.println(" The " + command + " Command is Executed Successfully. ");

pr.flush();

pr.close();

ou.close();

in.close();

}

catch(Exception e)

{

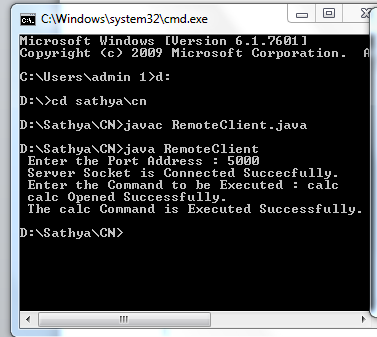
System.out.println(" Error : " + e.getMessage());

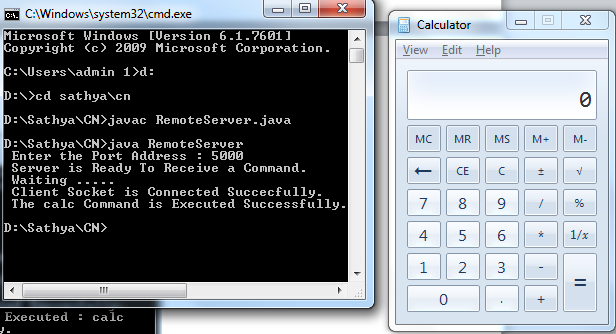
}

}

}

**OUTPUT**





**RESULT:**

Thus RPC using UDP has been implemented using JAVA Socket Programming